# Query-Adaptive Hash Code Ranking for Fast Nearest Neighbor Search

Tianxu Ji[†] Xianglong Liu[†*] Cheng Deng[‡] Lei Huang[†] Bo Lang[†]
[†]State Key Lab of Software Development Environment, Beihang University, Beijing, China
[‡]School of Electronic Engineering, Xidian University, Xi'an, China
{jitianxu, xlliu, huanglei, langbo}@nlsde.buaa.edu.cn chdeng.xd@gmail.com

## ABSTRACT

Recently hash-based nearest neighbor search has become attractive in many applications due to its compressed storage and fast query speed. However, the quantization in the hashing process usually degenerates its discriminative power when using Hamming distance ranking. To enable fine-grained ranking, hash bit weighting has been proved as a promising solution. Though achieving satisfying performance improvement, state-of-the-art weighting methods usually heavily rely on the projection's distribution assumption, and thus can hardly be directly applied to more general types of hashing algorithms. In this paper, we propose a new ranking method named QRank with query-adaptive bitwise weights by exploiting both the discriminative power of each hash function and their complement for nearest neighbor search. QRank is a general weighting method for all kinds of hashing algorithms without any strict assumptions. Experimental results on two well-known benchmarks MNIST and NUS-WIDE show that the proposed method can achieve up to 17.11% performance gains over state-of-the-art methods.

## Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; I.2.6 [**Artificial Intelligence**]: Learning

## Keywords

Hash Bit Weighting; Hash Code Ranking; Weighted Hamming Distance; Locality Sensitive Hashing

*corresponding author


*MM'14,* November 03–07, 2014, Orlando, FL, USA.

http://dx.doi.org/10.1145/2647868.2655018 .

## 1. INTRODUCTION

Hash based nearest neighbor search has attracted great attentions in many areas, such as large-scale visual search, data mining, pattern recognition and machine learning [4]. As the most well-known method, Locality-Sensitive Hashing (LSH) pioneered the hashing paradigm [1]. It projects and quantizes high-dimensional data to the binary hash code, and gains efficiency in terms of storage cost and query speed by computing the Hamming distance. Its theoretical analysis meanwhile ensures that nearest neighbors of each point are mapped to the same hash code with a high probability.

Due to the random generation of hash projections, LSH usually needs a quite large budget of hash functions to achieve desired discriminative power, which certainly brings large memory and time consumption. Many following work devote to pursuing compact hash codes in unsupervised [10,15] and supervised manner [5, 12]. Besides, there are a variety of powerful techniques to improve the compactness of hash codes like nonlinear hashing [6, 9], multiple features [12, 13], multiple bits [10] and bit selection [11].

The Hamming distance ranking helps achieve compressed storage and fast computation in hash-based nearest neighbor search. However, the quantization in hashing loses the exact ranking information among the samples, and thus degenerates the discriminative power of the Hamming distance measurement. For instance, in practice there exist more than one buckets that share the same Hamming distance to the query, and subsequently samples falling in these buckets will be ranked equally using Hamming distance. To improve the ranking accuracy using Hamming distance, it is necessary to enable fine-grained ranking by alleviating the quantization loss. One of the most powerful and successful techniques is hash bit weighting, which assesses the quality of each hash bit to improve the discriminative power of the hash code. Similar to traditional visual reranking [2], such process can be efficiently performed only on the top ranked results within certain Hamming radius.

Along this direction, [8] proposed a query-adaptive Hamming distance ranking method using the learned bitwise weights for a diverse set of predefined semantic concept classes. [17] studied a query-sensitive hash code ranking algorithm (QsRank) for PCA-based hashing algorithms without compressing query points to discrete hash codes. [16] also presented a state-of-the-art weighted Hamming distance ranking algorithm (WhRank) based on the data-adaptive and query-sensitive bitwise weight. Though achieving promising performance improvement, methods like WhRank heavily rely on the assumption of data distribution (eg., Laplace dis-

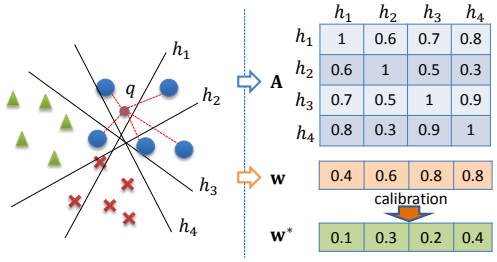



**Figure 1: Demonstration of the proposed method.**

tribution for spectral hashing [15]). Moreover, these methods are usually designed for projection based hashing algorithms. Therefore, they can hardly be directly applied for nonlinear hashing algorithms like spherical hashing [7] and K-means hashing [6].

In this paper, we propose a new ranking method with weighted hamming distance. By exploiting the similarity between the query and database samples, we learn a set of query-adaptive bitwise weights that characterize both the discriminative power of each hash function and their complement for nearest neighbor search. Assigning different weights to individual hash bit will distinguish the results sharing the same hamming distance, and obtain a more fine-grained and accurate ranking order. Compared to existing methods, our method is more general for different types of hashing algorithms, without strict assumptions on the data distribution. Meanwhile, it can faithfully enhance the overall discriminative power of the weighted Hamming distance.

The rest of the paper is organized as follows: The details of our approach are present in Section 2. Section 3 describes settings of our experiments and discusses the experimental results. Finally, we conclude in Section 4.

## 2. THE PROPOSED APPROACH

In this section, we propose the query-adaptive ranking method (named QRank) using weighted hamming distance. It utilizes the similarity relationship between the query point and its neighbors in the database to measure the overall discriminative power of the hash code, simultaneously considering both the quality of hash functions and their correlations.

### 2.1 Query-Adaptive Weight

Given a set of $n$ training samples $\{x_i \in \mathbb{R}^d, i = 1, \ldots, n\}$ and a set of $m$ hash functions $H(\cdot) = \{h_1(\cdot), \ldots, h_m(\cdot)\}$, each training sample $x_i$ is encoded into hash bit $y_{ik} = h_k(x_i) \in \{-1, 1\}$ by the $k$-th hash function. With the hash functions and corresponding weights $w$, the weighted Hamming distance between any two points $x_i$ and $x_j$ are usually defined as $d_h(x_i, x_j) = \sum_{k=1}^{m} w_k(y_{ik} \otimes y_{jk})$.

To improve the ranking precision of the weighted distance $d_h$, a data-dependent and query-adaptive $w_k$ should be learnt to characterize the overall quality of the $k$-th hash function in $H$. Intuitively, for a query point $q$, a hash function well preserving $q$'s nearest neighbors $\mathrm{NN}(q)$ (eg., $h_3$ and $h_4$ in Figure 1) should play a more important role in the weighted Hamming distance, and thus a larger weight should be assigned to it.

Formally, for a high-quality hash function $h_k$, if $p \in \mathrm{NN}(q)$, then the higher its similarity $s(p, q)$ to $q$, the larger the probability that $h_k(q) = h_k(p)$. Therefore, based on the neighbor preservation of each hash function, we define its weight using the spectral embedding loss [11, 15]:

$$\begin{aligned} w_k &= -\frac{1}{2} \sum_{p \in \mathrm{NN}(q)} s(q,p) \|h_k(q) - h_k(p)\|^2 \\ &= \sum_{p \in \mathrm{NN}(q)} s(q,p) h_k(q) h_k(p) + const. \end{aligned} \tag{1}$$

where we constrain $\sum_{p \in NN(q)} s(q, p) = 1$. Note that in the above definition the similarity can be adaptively tailored for different scenarios.

To make the weight positive and sensitive to the capability of neighbor preservation, in practice we use the following form with $\gamma > 0$:

$$w_k = \exp\left[\gamma \sum_{p \in \mathrm{NN}(q)} s(q,p) h_k(q) h_k(p)\right]. \tag{2}$$

In the above definition, one important question is how to efficiently find the query's nearest neighbors $\mathrm{NN}(q)$ at the online query stage. It is infeasible to find the exact ones among the whole database. One way to speedup the computation is choosing $n_l \ll n$ landmarks to represent the database using various techniques like K-means, where the approximated nearest neighbors can be quickly discovered by linear scan. We adopt the simple way by randomly sampling $n_l$ points as the landmarks at the offline training stage.

### 2.2 Weight Calibration

As Figure 1 demonstrates that hash function $h_3$ and $h_4$ shows satisfying capability of neighbor preservation for query $q$, but the large correlation between them indicates undesired redundancy between them. Instead, though function $h_2$ performs worse than $h_3$ and $h_4$, but it serves as a complement to $h_4$, with which they together can well preserve all neighbor relations of $q$. This observation motivates us to further calibrate the query-adaptive weights, taking the correlations among all hash functions into consideration.

Given any pair of hash functions $h_i$ and $h_j$ from $H$, if they behave similarly on a certain set of data points (i.e., the hash bits encoded by them are quite similar), then we can regard that the two hash functions are strongly correlated. In practice, to improve the overall discriminative power of the hash codes, uncorrelated (or independent) and high-quality hash functions should be given higher priority.

Since computing higher-order independence among hash functions is quite expensive, we approximately evaluate the independence based on the pair-wise correlations between them. Specifically, we introduce the mutual independence between hash functions based on the mutual information $\mathrm{MI}(y_i, y_j)$) between the bit variables $y_i$ and $y_j$ generated by hash function $h_i$ and $h_j$:

$$a_{ij} = \exp\left[-\lambda \mathrm{MI}(y_i, y_j)\right], \tag{3}$$

where $\lambda > 0$ and $a_{ij} = a_{ji}$, forming a symmetrical independence matrix $A = (a_{ij})$.

Then we calibrate the query-adaptive weights $w_k$ by reweighing it using a positive variable $\pi_k$. Namely, the new bitwise query-adaptive weight is given by

$$w_k^* = w_k \pi_k, \tag{4}$$

which should overall maximize both the neighbor preservation and the independence between hash functions. We formulate it as the following quadratic programming problem:

$$\max_{\pi} \quad \sum_{ij} w_i^* w_j^* a_{ij} \qquad s.t. \quad \mathbf{1}^T \pi = 1, \ \pi \succeq 0. \tag{5}$$

The above problem can be efficiently solved by a number of powerful techniques like replicator dynamics [11].

## 2.3 Data-Dependent Similarity

As aforementioned, the similarity between the query and database samples plays an important role in pursuing query-adaptive weights in Sec. 2.1. Since in practice data points are usually distributed on certain manifold, the standard Euclidean metric cannot strictly capture their global similarities. To obtain similarity measurement adaptive to different datasets, we adopt the anchor graph to represent any sample $x$ by $z(x)$ based on their local neighbor relations to anchors points $\mathcal{U} = \{u_k \in \mathbb{R}^d\}_{k=1}^r$, which can be generated efficiently by clustering or sampling [10]:

$$[z(x)]_j = \begin{cases} \frac{\mathcal{K}(x,u_j)}{\sum_{u'_j \in \mathrm{NN}(x)} \mathcal{K}(x,u_{j'})}, & \text{if } u_j \in \mathrm{NN}(x) \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

where $\mathrm{NN}(x)$ denotes $x$'s nearest anchors in $\mathcal{U}$ according to the predefined kernel function $\mathcal{K}(x, u_j)$ (e.g., Gaussian kernel). The highly sparse $z(x)$ serves as a nonlinear and discriminative feature representation, and can be used to efficiently approximate the data-dependent similarities between samples. Specifically, for query $q$ and any point $p$, their similarity can be computed by

$$s(p, q) = \exp(-\|z(p) - z(q)\|^2/\sigma^2), \tag{7}$$

where $\sigma$ is set to the largest distance between $z(p)$ and $z(q)$.

## 2.4 Online Query

At the offline stage, both $n_l$ sampled landmarks and $r$ anchors can be obtained efficiently, and the landmarks can be represented using anchors in $O(n_l r)$. At the online stage, first the query feature is transformed into anchor representation and used to compute its similarities to the landmarks in $O(r+n_l)$, second the query-adaptive weights can be obtained by quadratic programming (5) in polynomial time, and finally the results are fast ranked according to the weighted hamming distances.

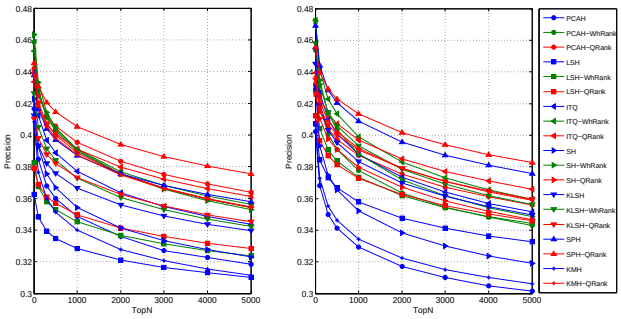
# 3. EXPERIMENTS

To evaluate the proposed query-adaptive hash code ranking method (QRank), we conduct extensive experiments on the real-world image datasets MNIST and NUS-WIDE:

MNIST: it includes 70K 784 dimensional images, each of which is associated with a digit label from '0' to '9'.

NUS-WIDE: it is one of largest real-world labeled image datasets, consisting of 270K images with 81 ground truth concept tags, and we consider 25 most frequent tags.

We compare QRank with the state-of-the-art binary codes ranking method WhRank [16] over a variety of baseline hashing algorithms including Locality Sensitive Hashing (LSH) [1], Spectral Hashing (SH) [15], Kernelized Locality-Sensitive Hashing (KLSH) [9], PCA-Hashing (PCAH) [14], Iterative Quantization (ITQ) [3], Spherical Hashing (SPH) [7], K-means Hashing (KMH) [6]. Since WhRank outperformed QsRank [17] significantly as reported in [16], we omit the direct comparison between our QRank and QsRank.




(a) Precision @ 48 bits (b) Precision @ 96 bits

**Figure 3: Performances comparison on NUS-WIDE.**

We adopt the popular performance metric including precision, recall and mean average precision (MAP) in our experiments. All results are averaged over 10 independent runs to suppress the randomness. For each run, we randomly sample 5,000 images as the training data, 3,000 as the landmarks and 1,000 as query images. The true nearest neighbors are defined as the images sharing at least one common tag.

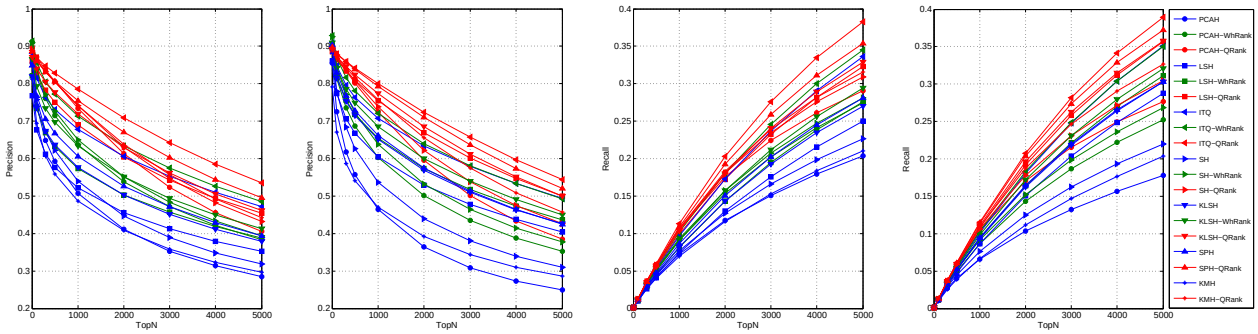
## 3.1 Results and Discussions

Figure 2 shows the precision and recall curves respectively using 48 and 96 bits on MINST. We can easily find out that both hash bit weighting methods (WhRank and QRank) achieved better performances than the baseline hashing algorithms. The observation indicates that the hash code ranking based on weighted Hamming distance hopefully serves as a promising solution to boosting the performance of hash-based retrieval. In all cases, our proposed QRank consistently achieves the superior performances to WhRank over different hashing algorithms. Figure 4(a) depicts the overall performance evaluation using MAP with 96 hash bits, where we get a similar observation that QRank outperforms WhRank significantly, e.g., compared to WhRank, QRank obtains 17.11%, 11.36%, 14.06% and 11.39% performance gains respectively over LSH, ITQ, SH and KLSH.

Besides MNIST, Figure 3 and 4(b) present the precision curves and MAP on the NUS-WIDE dataset when using 48 and 96 bits. By comparing the performance of QRank with that of WhRank and baseline hashing algorithms, we get a similar conclusion as on MNIST that both QRank and WhRank can enhance the discriminative power of hash functions by weighting them elegantly, and meanwhile in all cases QRank clearly outperforms WhRank owing to its comprehensive capability of distinguishing high-quality functions.

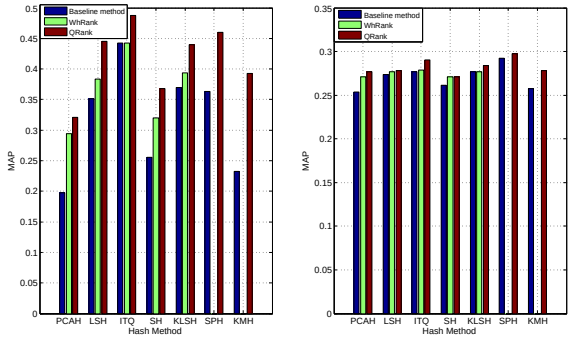
It is worth noting that since WhRank depends on the distribution of the projected samples, it cannot be directly applied to hashing algorithms like SPH and KMH. Instead, the proposed method derives the bitwise weight only relying on the data-dependent similarity. Therefore, it can not only capture the neighbor relations of the query, but also possess universality suiting for all hashing algorithms. Figure 2-4 show that QRank obtains significant performance gains over SPH and KMH in all cases.

We investigate the effect of different parts of QRank in Table 1 by comparing the performance of QRank with or without the weight calibration (named QRank$^-$). The results are reported over LSH, SH, PCAH and ITQ respectively using 96 bits on MNIST. It is obvious that QRank$^-$




(a) Precision @ 48 bits (b) Precision @ 96 bits (c) Recall @ 48 bits (d) Recall @ 96 bits

**Figure 2: Performances comparison on MNIST.**




(a) MNIST @ 96 bits (b) NUS-WIDE @ 96 bits

**Figure 4: MAP (%) on MNIST and NUS-WIDE.**

**Table 1: MAP (%) of different parts of QRank on MNIST**

| Ranking Method | LSH | SH | PCAH | ITQ |
|---|---|---|---|---|
| Baseline | 35.53 | 25.91 | 19.87 | 44.14 |
| QRank$^-$ | 40.71 | 31.39 | 22.07 | 46.87 |
| QRank | **44.77** | **37.02** | **32.32** | **49.15** |

only using query-adaptive weights without considering the redundance among hash functions is able to boost the hashing performance, but QRank appended with the weight calibration can further bring significant (up to 46.44%) performance gains. This indicates that both the individual quality of each hash function and their complement are critical for fine-grained neighbor ranking.

Finally, we present the time cost of different methods in Table 2. We can see that though our QRank spends more time than WhRank on learning query-adaptive weight (50.19 ms), on the whole this part is relatively small and the online query is quite efficient in practice.

## 4. CONCLUSIONS

As described in this paper, we proposed a new hash code ranking method named QRank, which learns a query-adaptive bitwise weights by simultaneously considering both the individual quality of each hash function and their complement for nearest neighbor search. Compared to state-of-the-art weighting methods, QRank serves as a general solution to enabling fine-grained ranking over all kinds of hashing algorithms. Due to both the data-dependent similarity and query-adaptive weights, QRank significantly and efficiently boosts the ranking performance in practice.

**Table 2: Time cost (ms) of different methods on MNIST**

| | Baseline Hash | WhRank | QRank |
|---|---|---|---|
| weight computing | 0 | 1.27 | 50.19 |
| distance ranking | 39.18 | 141.15 | 141.15 |
| Total time | 39.18 | 142.42 | 191.34 |

## 5. ACKNOWLEDGMENTS

This work is supported in part by NSFC 61370125 and 61101250, YWF-14-JSJXY-010, NCET-12-0917, SKLSDE 2013ZX-05 and 2014ZX-07.

## 6. REFERENCES

[1] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *SCG*, 2004.

[2] C. Deng, R. Ji, W. Liu, D. Tao, and X. Gao. Visual reranking through weakly supervised multi-graph learning. In *IEEE ICCV*, 2013.

[3] Y. Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *IEEE CVPR*, pages 817 –824, june 2011.

[4] J. He, J. Feng, X. Liu, T. Cheng, T.-H. Lin, H. Chung, and S.-F. Chang. Mobile product search with bag of hash bits and boundary reranking. In *IEEE CVPR*, 2012.

[5] J. He, W. Liu, and S.-F. Chang. Scalable similarity search with optimized kernel hashing. In *ACM SIGKDD*, 2010.

[6] K. He, F. Wen, and J. Sun. K-means hashing: An affinity-preserving quantization method for learning binary compact codes. In *CVPR*, 2013.

[7] J. Heo, Y. Lee, J. He, S.-F. Chang, and S. Yoon. Spherical hashing. In *CVPR*, 2012.

[8] Y.-G. Jiang, J. Wang, and C. Shih-Fu. Lost in binarization: query-adaptive ranking for similar image search with compact codes. In *ACM ICMR*, 2011.

[9] B. Kulis and K. Grauman. Kernelized locality-sensitive hashing for scalable image search. In *IEEE ICCV*, 2009.

[10] W. Liu, J. Wang, S. Kumar, and S.-F. Chang. Hashing with graphs. In *ICML*, 2011.

[11] X. Liu, J. He, B. Lang, and S.-F. Chang. Hash bit selection: a unified solution for selection problems in hashing. In *IEEE CVPR*, 2013.

[12] X. Liu, J. He, D. Liu, and B. Lang. Compact kernel hashing with multiple features. In *ACM MM*, 2012.

[13] J. Song, Y. Yang, Z. Huang, H. T. Shen, and R. Hong. Multiple feature hashing for real-time large scale near-duplicate video retrieval. In *ACM MM*, 2011.

[14] J. Wang, S. Kumar, and S.-F. Chang. Semi-supervised hashing for scalable image retrieval. In *IEEE CVPR*, 2010.

[15] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, 2008.

[16] L. Zhang, Y. Zhang, J. Tang, K. Lu, and Q. Tian. Binary code ranking with weighted hamming distance. In *CVPR*, 2013.

[17] X. Zhang, L. Zhang, and S. Heung-Yeung. Qsrank: Query-sensitive hash code ranking for efficient $\epsilon$-neighbor search. In *CVPR*, 2012.